Semantic-Preserving Adversarial Example Attack against BERT

Chongyang Gao[1*] **Kang Gu**[2*] **Soroush Vosoughi**[2] **Shagufta Mehnaz**[3]
[1]Northwestern University [2]Dartmouth College [3]Penn State University

Abstract

Adversarial example attacks against textual data have been drawing increasing attention in both the natural language processing (NLP) and security domains. However, most of the existing attacks overlook the importance of semantic similarity and yield easily recognizable adversarial samples. As a result, the defense methods developed in response to these attacks remain vulnerable and could be evaded by advanced adversarial examples that maintain high semantic similarity with the original, non-adversarial text. Hence, this paper aims to investigate the extent of textual adversarial examples in maintaining such high semantic similarity. We propose *Reinforce* attack, a reinforcement learning-based framework to generate adversarial text that preserves high semantic similarity with the original text. In particular, the attack process is controlled by a reward function rather than heuristics, as in previous methods, to encourage higher semantic similarity and lower query costs. Through automatic and human evaluations, we show that our generated adversarial texts preserve significantly higher semantic similarity than state-of-the-art attacks while achieving similar attack success rates (outperforming at times), thus uncovering novel challenges for effective defenses.

1 Introduction

In this paper, we focus on the generation of semantic-preserving adversarial examples. Table 1 displays two instances of adversarial examples for an original sentence where the NLP classification task is labeling reviews as positive or negative. Both adversarial examples were generated by replacing the highlighted words in Table 1 and successfully forced the model to change its prediction from positive to negative. However, the first adversarial example that replaces “like" with “hate" should not be considered an adversarial example because a human may also think that it is a negative review. On the contrary, the second adversarial example is more semantically similar to the original text, and a human may expect the review to be classified as positive, whereas the model is tricked into predicting the review as negative. Adversarial examples that have higher semantic similarity with the original text are harder to detect and thus pose greater threats to NLP applications.

Table 1: Difference between Poor and Tricky Adversarial Examples (AE) for an NLP Application

Original	I like this movie, she is a good actress		Prediction: Positive
Poor AE	I hate this movie, she is a good actress	like → hate	Prediction: Negative
Tricky AE	I like some movie, she is a good actress	this → some	Prediction: Negative

State-of-the-art attacks for generating textual adversarial examples typically consist of the following steps: (1) finding vulnerable words and ranking them, and (2) replacing the words one by one to generate adversarial samples. Li et al. (Li et al., 2020) rank the vulnerability of words by masking all words in the input sentence one at a time and then comparing the corresponding predictions' probabilities of the masked sentences. As for the second step, the lexical substitute models (Zhou et al., 2019) are used to generate adversarial examples. However, there are two significant drawbacks to the above framework: (i) the word importance ranking via masking ignores the correlations between words, (ii) the entire attack process relies on the synonym dictionary (Mrkšić et al., 2016) to constrain the replacement, which doesn't actively optimize adversarial examples to preserve semantic similarity. In this paper, we aim to extend textual adversarial attacks with the goal of increasing the semantic similarity between the original text and the generated adversarial example. This work has the potential to spur further research in this domain of problems and thus facilitate the development of advanced defense mechanisms.

First, we investigate the effects of computing

[*]Both authors contributed equally to this research.

word importance ranking via LIME (Ribeiro et al., 2016) to consider the information of multiple words. Recently, interpretability tools (Ribeiro et al., 2016; Lundberg and Lee, 2017) have been explored in membership inference attacks (Shokri et al., 2021) as well as generating (Liu et al., 2024, 2021) or detecting (Fidel et al., 2020) visual adversarial examples. We show that simply switching from masking to LIME can improve the attack performance noticeably. Secondly, to enforce the semantic similarity between the original text and generated adversarial examples, we introduce a reinforcement learning (RL)-based framework, namely, *Reinforce* attack. RL has previously been applied to reading comprehension (Hu et al., 2018), question answering (Yang et al., 2021), and sentence simplification (Zhang and Lapata, 2017). More specifically, we recast the attack process as a sequence tagging problem, where an agent is trained to identify vulnerable words for substitution to maximize a reward function that optimizes *four key metrics*: semantic similarity, attack success rate, input perturbation rate, and number of queries. We conduct extensive experiments on four classification datasets and one regression dataset to demonstrate the effectiveness of our attack methods. The contribution of this paper is twofold:

- We show the potential of using an interpretability tool (LIME (Ribeiro et al., 2016)) in the word importance ranking step that can produce a more accurate word ranking, thus improving the attack performance.
- We develop a reinforcement learning (RL)-based textual adversarial example generation attack dubbed as *Reinforce* attack that preserves higher semantic similarity between the original text and adversarial examples.

2 LIME Attack

Our key idea is that the explanations of LIME can be leveraged to identify words that are vulnerable to adversarial attacks. Instead of considering each word one by one as in previous work for finding vulnerable words (Li et al., 2020; Jin et al., 2020), LIME first generates neighborhood samples by randomly removing several words from the input sentence and querying the BERT to get output logits for each neighborhood sample. Then, a weighted linear model is learned by taking logits as the labels to approximate the locality of the prediction. The word importance is calculated by solving the weights of the linear model to minimize the sum of cosine distance between the logits of the original instance and neighborhood samples. Hence, LIME takes contextual information into account and scores each word's importance in a holistic way. More details are in Appendix A.

Algorithm 1 summarizes our adversarial example generation steps. The first step is to pre-process the text S and feed it into $LIME(\cdot)$ to obtain the important words. $LIME(\cdot)$ returns a ranked word list and we consider only the first q words from the ranked list, which is represented by I. After we acquire the list of the important words, we use a word replacement strategy as shown in Algorithm 1 to generate the adversarial examples. For each important word $w_j \in I$, we leverage BERT to identify the list of K candidates P^j. Let P be the list of all such P^js—representing the top-K candidates for all words in I. Note that, for every candidate in P, we filter P^j by a set of stop words. The attack is successful when the target model returns a label other than Y for the perturbed text S'. If the attack is not successful in a certain iteration, the next word is perturbed, and we check again for adversarial example success. Algorithm 1 sets the maximum perturbation rate at 0.25.

3 Reinforce Attack

Our key observation from state-of-the-art attacks is that none of these attacks optimizes for semantic similarity, which is a key metric for evaluating adversarial examples. Therefore, we incorporate the above illustrated adversarial examples generation into our RL-based framework, dubbed as **Reinforce** attack as in Figure 1, which optimizes the trade-offs among all the four key metrics during the attack process, i.e., attack success rate, semantic similarity, query number, and perturbation rate.

3.1 Key Metrics

Attack Success: The success rate is the main metric for evaluating the performance of the adversarial attack.

$$r^A = max(p_{ori} - p_{adv}, 0) \quad (1)$$

where p_{ori} is the original probability of the predicted class and p_{adv} is the resulting probability of adversarial sample.

Semantic Similarity: We consider the Universal Sentence Encoder (USE) (Cer et al., 2018) as another vital metric to evaluate semantic similarity

Algorithm 1 Adversarial example generation

Require: $S = [w_0, w_1, ..., w_n]$
$Y \leftarrow$ ground-truth label of sentence S
$l \leftarrow 0.25 \times n$ //Maximum number of word substitutions
$LIME(\cdot) : S \rightarrow [w_i, ...]$ //The length of $[w_i, ...]$ is q
$Logit(\cdot) : S \rightarrow \mathbb{R}^C$ //C is the number of classes
Ensure: S_{adv} //Adversarial example
$I = [w_i, ...] \leftarrow LIME(S)$ //q important words in descending order
$P^{\in q \times K}$ = top-K candidates for all words in I using BERT
$n_s = 0$ //Number of substituted words
for w_j in I **do**
  if $n_s > l$ **then**
    return False //Fail to generate adversarial example
  else
    for P_k^j in P^j **do**
      $S' = [w_0, w_1, ..., w_{j-1}, P_k^j, ...]$
      if $argmax(Logit(S'))! = Y$ **then**
        return $S_{adv} = S'$ //Attack successful
      else
        if $Logit(S')[Y] < Logit(S_{adv})[Y]$ **then**
          $S_{adv} = S'$ //Update S_{adv}
          $n_s += 1$
    end for
end for

directly, which is widely used to calculate the similarity between a pair of texts. r^S represents the output score of USE.

$$r^S = USE(S, S_{adv}) \tag{2}$$

where S and S_{adv} are the original and adversarial sentences, respectively.

Query Number: The query number reflects the efficiency of the attack. While the attack reward r^A tries to encourage the model to generate misleading samples, the query reward r^Q ensures that the attack success is not achieved at the cost of a high number of queries.

$$r^Q = \frac{Q}{n} \tag{3}$$

where Q is the number of queries and n is the length of the sentence.

Perturbation Rate: We expect the attack to succeed by replacing a minimal number of words. The reward r^P simply calculates the perturbation rate to regularize the reward function.

$$r^P = \frac{P}{n} \tag{4}$$

where P is the number of perturbed words and n is the length of the sentence.

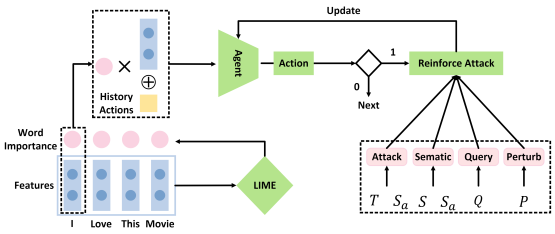

Figure 1: Reinforce attack framework. T is the target model, S and S_a are original and adversarial sentences, respectively, Q is the query number, and P represents perturbation rate. Note that, in practice, we use the sorted words according to the weights.

4 Experiments

Dataset Description: We apply our method to both classification and regression tasks. The datasets used in our experiments for classification are Yelp (Yelp, 2021), IMDB (IMDB, 2018), AG's News (AG, 2019), and FAKE (FAKE, 2018). For regression, we use Blog Authorship Corpus ((Santosh et al., 2013)). We follow the configuration in (Li et al., 2020) to test on 1000 samples, which are the same splits used by (Jin et al., 2020). As

Table 2: Comparison of our attacks (*LIME attack* and *Reinforce attack*) with existing work.

Classification Task

Dataset	Attack Method	Avg Len	Original Acc	After Attack Acc	Perturb %	Query	Semantic Sim	Cosine Sim
IMDB	GA (Alzantot et al., 2018)	215	90.9	45.7	4.9	6493	-	-
	TextFooler (Jin et al., 2020)			13.6	6.1	1134	0.86	-
	BERT-Attack (Li et al., 2020)			11.4	4.4	454	0.86	0.87
	LIME Attack (Ours)			4.1	**3.0**	742	0.80	0.91
	Reinforce Attack (Ours)			**1.9**	3.3	**367**	**0.97**	**0.94**
Yelp	GA (Alzantot et al., 2018)	157	95.6	31.0	10.1	6137	-	-
	TextFooler (Jin et al., 2020)			6.6	12.8	743	0.74	-
	BERT-Attack (Li et al., 2020)			**5.1**	**4.1**	**273**	0.77	0.85
	LIME Attack (Ours)			6.1	4.7	352	0.86	0.84
	Reinforce Attack (Ours)			6.2	10.8	360	**0.96**	**0.88**
Fake	GA (Alzantot et al., 2018)	885	97.8	58.3	1.1	28508	-	-
	TextFooler (Jin et al., 2020)			19.3	11.7	4403	0.76	-
	BERT-Attack (Li et al., 2020)			15.5	**1.1**	**1558**	0.81	0.88
	LIME Attack (Ours)			6.0	4.0	2981	0.65	0.72
	Reinforce Attack (Ours)			**2.6**	4.4	2811	**0.98**	**0.92**
AG	GA (Alzantot et al., 2018)	43	94.2	51.0	16.9	3495	-	-
	TextFooler (Jin et al., 2020)			12.5	22.0	357	0.57	-
	BERT-Attack (Li et al., 2020)			**10.6**	15.4	213	0.63	0.71
	LIME Attack (Ours)			16.2	18.3	387	0.81	0.75
	Reinforce Attack (Ours)			15.0	**15.1**	**210**	**0.94**	**0.85**

Regression Task

Dataset	Method	Avg Len	Original MAE	Attacked MAE	Perturb %	Query	Semantic Sim	Cosine Sim
Blog	BERT-Attack (Li et al., 2020)	195	6.5	10.5	**2.0**	**151**	0.95	0.70
	Reinforce Attack (Ours)		-	**14.0**	3.9	199	**0.97**	**0.86**

for regression, we randomly split a subset of 1000 random samples from the dataset for testing.

Setup of Automatic Evaluation: To measure the quality of the generated samples comprehensively, we set up extensive automatic evaluation metrics as in (Li et al., 2020). The attack accuracy, which is the accuracy of the target model on adversarial samples, is the core metric measuring the effectiveness of the attack model. In addition, the perturbation rate is also vital since less perturbation usually means more semantic consistency. Furthermore, the query number per sample is a key metric, reflecting the attack model's efficiency. Finally, we also use the Universal Sentence Encoder to measure the semantic similarity between the original sentence and the adversarial sample.

Experiment Results: We compare our Reinforce attack and LIME attack, which is the version without using reinforcement framework, with three existing works: GA (Alzantot et al., 2018), TextFooler (Jin et al., 2020), and BERT-Attack (Li et al., 2020). The target model is BERT-base in this section.

Classification: As shown in Table 2, both our LIME attack and Reinforce attack achieve comparable or even better results compared to the other attack methods. Our Reinforce attack achieves an average after-attack accuracy of about 6.4%, which is a significant improvement compared to the BERT-Attack (10.6%) and LIME attack (8.1%). We also observe that methods with LIME perform better on datasets with longer average lengths (IMDB and Fake). Most notably, Reinforce attack consistently outperforms other attack methods in terms of semantic similarity by a large margin. The semantic similarity reward in Reinforce attack plays a vital role in maintaining high semantic consistency throughout the attack process.

Regression: Currently, LIME only supports explaining classification tasks because LIME relies on the prediction probabilities to solve the explanations. To resolve the issue, the regression task needs to be discretized into the classification task. Therefore, we only compare the vanilla BERT-Attack and our Reinforce attack. Reinforce attack achieves an attacked MAE of 14.0, outperforming the BERT-Attack by $\sim 33\%$.

5 Conclusions and Future Work

We develop and evaluate *Reinforce* attack that generates successful adversarial texts while preserving the original text's semantics. We believe that this unveils emerging challenges to make NLP applications more secure and robust. In the future, we aim to evaluate existing defenses against such semantic similarity-preserving adversarial examples and develop more robust defenses against these attacks.

A Important Words Selection

To obtain the important words, we construct a function that takes the text as input and calls the target BERT model to generate the logit probability for each class as output. Then LIME employs the constructed function to predict the importance of all words. Specifically, LIME first randomly masks the words in the original sentence and then uses the language model to get the logit probability of the masked sentence. The LIME algorithm trains a ridge regression model by minimizing the sum of cosine distance between the logits of the original sentence and its variations to estimate the importance of local words. Then, we can have the ranking list of the words II.

Here is a simple example of how LIME measures the importance of words[1]. Suppose the black box model is a decision tree trained on a document word matrix and aims to classify YouTube comments as spam (1) or normal (0). To explain "*For Christmas Song visit my channel! ;)*" with label 1, LIME generates some random variations of the sample, which will be used to train the local linear model. As in Table 3, each column corresponds to one word in the sentence and each row is a variation with 1/0 representing the existence/absence of the word. The "PROB" column shows the predicted probability of spam resulting from each variation. The "WEIGHT" column shows the proximity of the variation to the original sentence, calculated as 1 minus the proportion of words that are removed. For example, if 1 of 7 words was removed, the proximity is 1 - 1/7 = 0.86. The LIME algorithm then trains a linear model by minimizing the sum of the cosine distance between the logits of the original sentence and its variations to estimate the local word importance. In this example, LIME finds that the word "channel" has a high probability of spam. Since the rest of the words have no impact on the prediction, their weights will be estimated as nearly zero.

B Human Evaluation

Since the similarity metrics may not agree with human intuition, we perform a human evaluation to evaluate further the generated adversarial examples via Amazon Turk. We use the IMDB and Blog datasets for evaluation. There are 50 original samples, 50 corresponding adversarial samples generated by BERT-Attack, and 50 samples generated by our methods, which are randomly selected for each dataset. Firstly, we ask the annotators to rate the grammaticality of the sentences from 1 to 5 (5 being the best), following (Li et al., 2020). Secondly, we ask the annotators to compare the semantic similarity of reference sentences with those generated by the attack methods. The scale is 0 to 1, where 1 is similar, 0 is dissimilar and 0.5 is the middle, following (Jin et al., 2020). Thirdly, the human workers are asked to decide whether the generated samples' labels are consistent with the original sentences' labels. If the labels are the same, then the score is 1. Otherwise, the score is 0. The sentiment of the original sentence is compared to itself, so the label consistency score of original sentences is 1. As shown in Table 4, both our LIME attack and Reinforce attack outperform the BERT-Attack in the IMDB dataset.

[1] https://christophm.github.io/interpretable-ml-book/lime.html#lime-for-text

References

AG. 2019. https://www.kaggle.com/amananandrai/ag-news-classification-dataset.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. In *arXiv:1803.11175*.

FAKE. 2018. https://www.kaggle.com/c/fake-news/data.

Gil Fidel, Ron Bitton, and Asaf Shabtai. 2020. When explainability meets adversarial learning: Detecting adversarial examples using shap signatures. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*.

IMDB. 2018. https://datasets.imdbws.com.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Table 3: Variations of Text Sample

For	**Christmas**	**Song**	**visit**	**My**	**channel!**	**;)**	PROB	WEIGHT
1	0	1	1	0	0	1	0.17	0.57
0	1	1	1	1	0	1	0.17	0.71
1	0	0	1	1	1	1	0.99	0.71
1	0	1	1	1	1	1	0.99	0.86
0	1	1	1	0	0	1	0.17	0.57

Table 4: Human evaluation results on IMDB and Blog.

Dataset		Semantic	Grammar	Label Consistency
IMDB	Original	1	3.39	1
	BERT-Attack (Li et al., 2020)	0.82	3.24	0.88
	LIME Attack (Ours)	0.81	3.44∗	0.90
	Reinforce Attack (Ours)	0.87∗	3.31	0.93∗
Blog	Original	1	3.51	-
	Reinforce Attack (Ours)	0.80	3.09	-

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Ninghao Liu, Mengnan Du, Ruocheng Guo, Huan Liu, and Xia Hu. 2021. Adversarial attacks and defenses: An interpretation perspective. *ACM SIGKDD Explorations Newsletter*, 23(1):86–99.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, et al. 2024. Best practices and lessons learned on synthetic data for language models. *arXiv preprint arXiv:2404.07503*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

K Santosh, Romil Bansal, Mihir Shekhar, and Vasudeva Varma. 2013. Author profiling: Predicting age and gender from blogs. In *CLEF*.

Reza Shokri, Martin Strobel, and Yair Zick. 2021. On the privacy risks of model explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 231–241.

Xu Yang, Chongyang Gao, Hanwang Zhang, and Jianfei Cai. 2021. Auto-parsing network for image captioning and visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2197–2207.

Yelp. 2021. https://www.yelp.com/dataset.

X. Zhang and M. Lapata. 2017. Sentence simplification with deep reinforcement learning. In *EMNLP*.

W Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou. 2019. Bert-based lexical substitution. In *ACL*.